The CLPZINC modeling language and its long compilation chain to solving

Philippe Morignot

Constraint Satisfaction Problems (CSP)

- A paradigm for solving combinatorial problems
- Finite-domain variables, constraints.

$$\forall i \in [1, 30],\ x_i \in [1, 200],\ y_i \in [1, 200]$$

$$\forall i, j,\ i < j : x_i + i \le x_j \ \vee\ x_j + j \le x_i \ \vee\ y_i + i \le y_j \ \vee\ y_j + j \le y_i$$

- Reification: $\begin{cases} x_1 + 3 \le x_2 \text{ satisfied: bool } b = 1 \\ x_1 + 3 \le x_2 \text{ violated: bool } b = 0 \end{cases}$
- Search strategy: additional constraints which orient search and propagation of the solver.
  - e.g., labelling: enumerating the values, one by one.

ZINC

- A high-level language for stating a CSP
- Zinc specification and the reduced MiniZinc implementation (NICTA).
- <u>Example:</u> Korf.

int: n = 30;

int: range = 200;

array[1..n] of var 1..range: x;

array[1..n] of var 1..range: y;

constraint forall(i in 1..n-1, j in i+1..n)

(x[i] + i <= x[j] $\vee$ ***x[j] + i <= x[i]*** $\vee$ ***y[i] + i <= y[j]*** $\vee$ ***y[j] + j <= y[i]);***

solve satisfy;

- FlatZinc: A low-level language easily parsed by solvers (e.g., CHOCO, JaCoP, SICTUS). A MiniZinc model is compiled into a FlatZinc one.

  MiniZinc -> FlatZinc -> (solvers)

- ***solve satisfy;*** ***solve minimize X;*** ***solve maximize Y;***

Weakness of ZINC

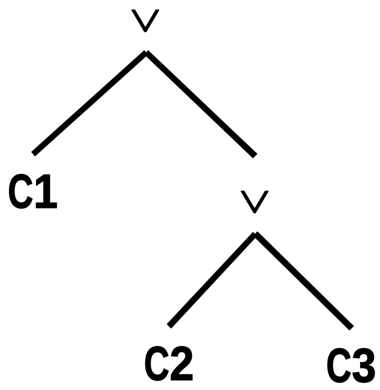
```
int: n = 30;
int: range = 200;
array[1..n] of var 1..range: x;
array[1..n] of var 1..range: y;
constraint forall(i in 1..n-1, j in i+1..n)
        (x[i] + i <= x[j]   \/   x[j] + i <= x[i]   \/   y[i] + i <= y[j]   \/   y[j] + j <= y[i]);
solve :: seq_search ([
        int_search(x, input_order, indomain_min, complete),
        int_search(y, first_fail, indomain_split, complete)
]) satisfy;
```

- Zinc’s annotations express little only of search strategies …

CLPZINC

- Reified constraints express disjunctions

(C1 ; (C2 ; C3))



var 0..1 : XX1;

var 0..1 : XX2;

*constraint XX1 = 0 -> **C1**;*

*constraint XX1 = 1 /\ XX2 = 0 -> **C2**;*

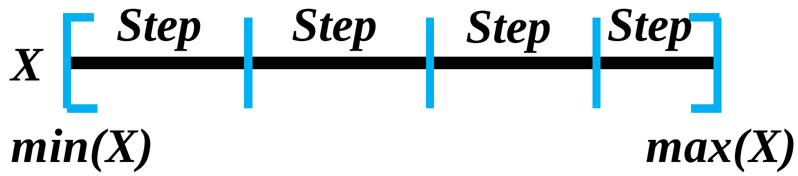
*constraint XX1 = 1 /\ XX2 = 1 -> **C3**;*

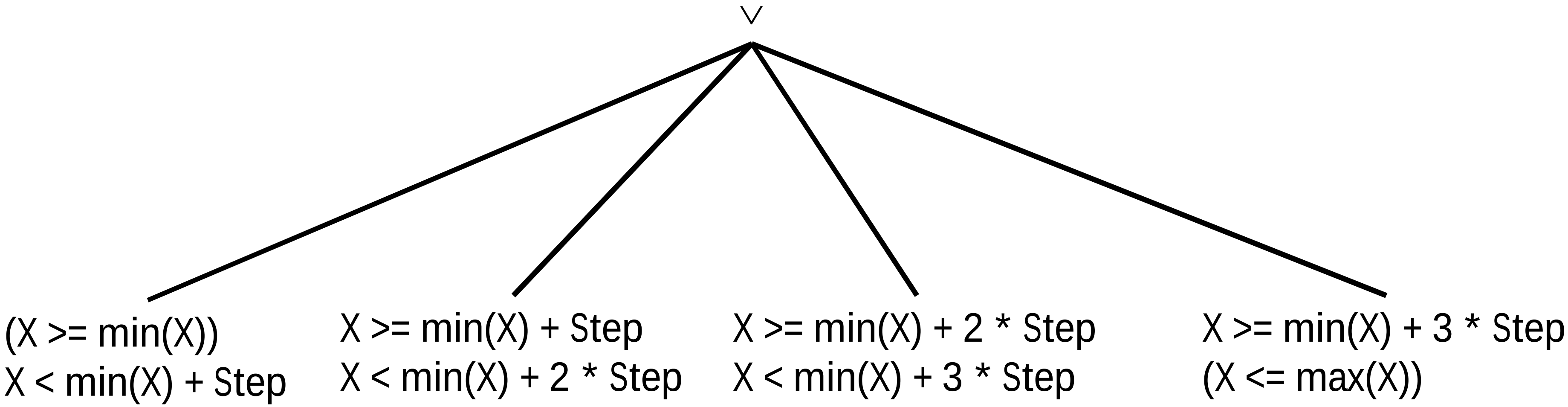
constraint XX1 = 0 -> XX2 = 0;

- CLPZINC is a Horn-clause extension of MiniZinc for expressing search strategies in CLP clauses.
  – e.g., labelling: ***between(A, X, B) :-***
  A <= B, (X = A ; S = A + 1, between(S, X, B)).
- Extended with records, forall, exists

Example of search strategies

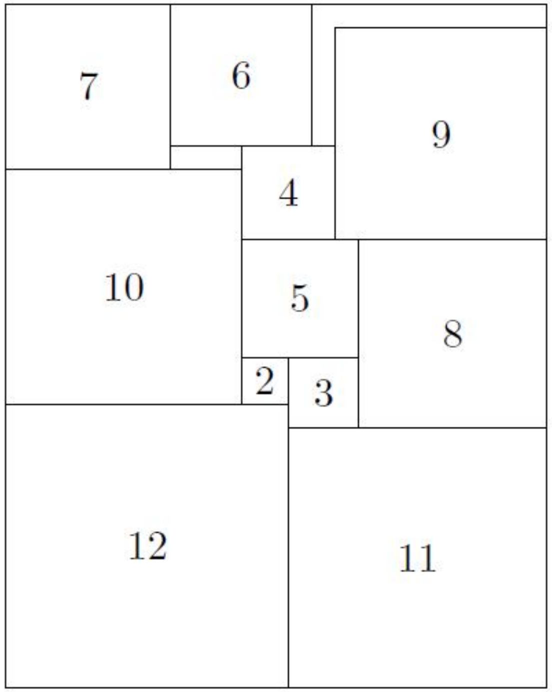
```
interval_splitting(X, Step, Min, Max) :-
  Min + Step <= Max,
  NextX = min(X) + Step,
  (   X < NextX ;
      X >= NextX,
      interval_splitting(X, Step, Min + Step, Max) ).
interval_spliting(X, Step, Min, Max) :-
  Min + Step > Max.
```





Packing

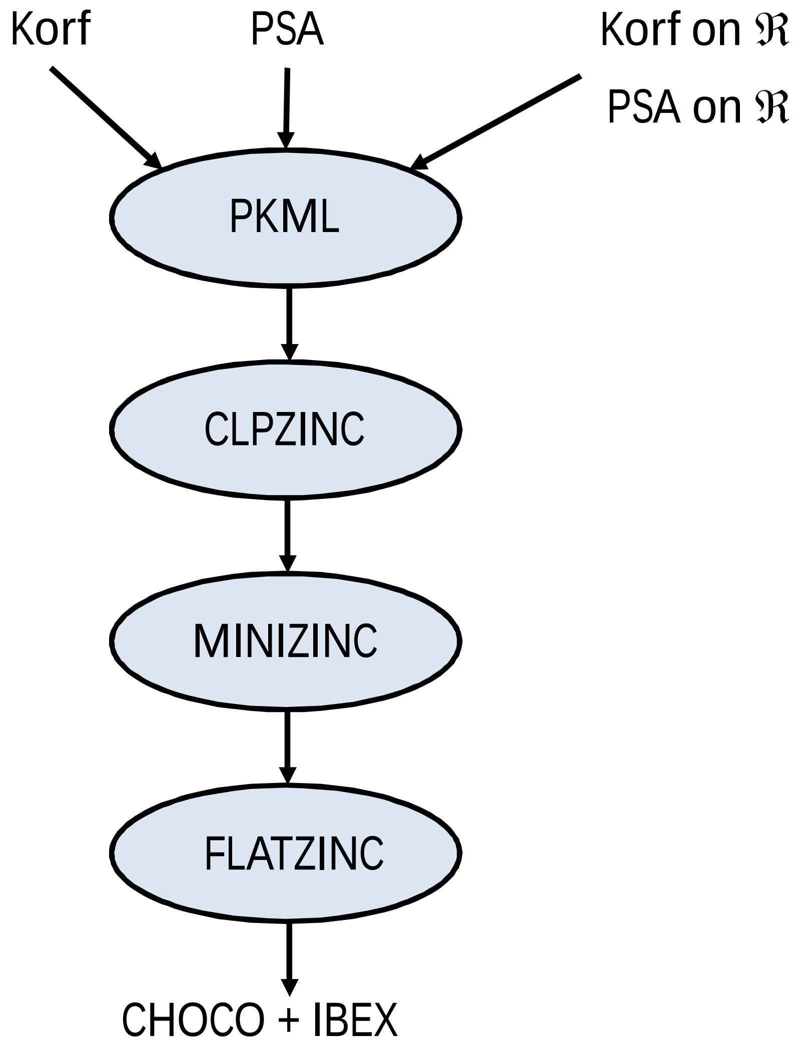
- Optimal packing of objects with complex shapes
- Allen, RCC, PKML libraries



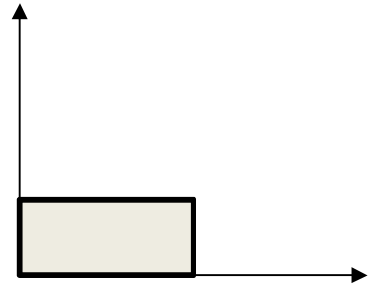
Korf, n = 12

- Applications:
  - Korf: How to place *n* non-overlapping squares, of size *1x1* up to *nxn*, in a minimal surface?
  - Real data from the company PSA

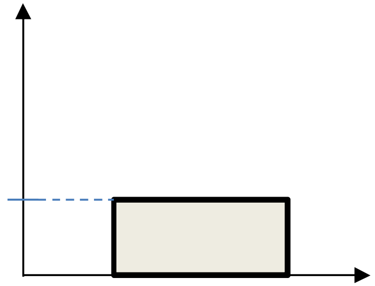
Compilation chain



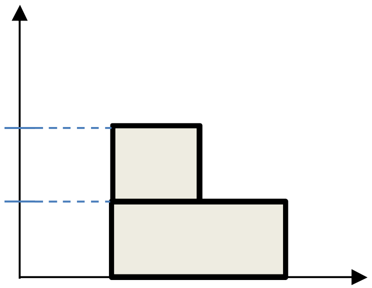
Packing Knowledge Modelling Language (PKML)



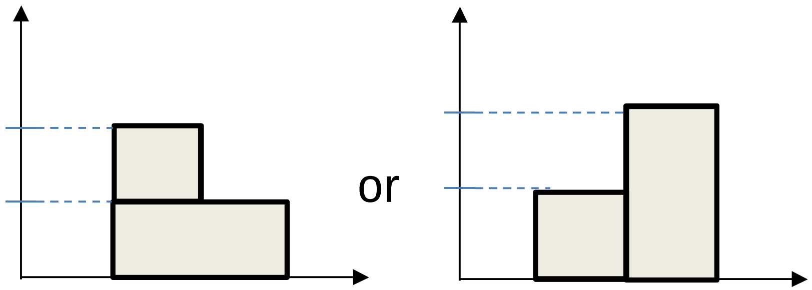
Box



Shifted Box

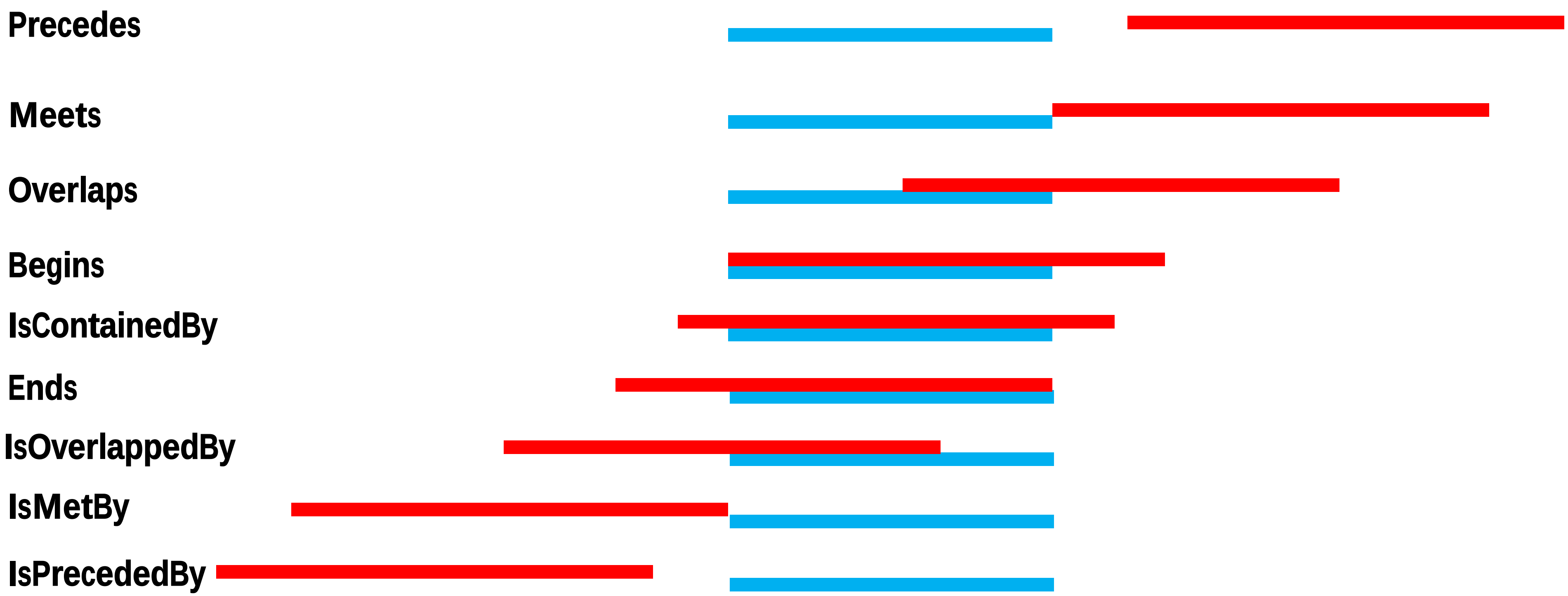


Shape



Object

Allen's relations between time intervals



precedes(T1, T2, D) :-
T1.end[D] < T2.start[D].
meets(T1, T2, D) :-
T1.end[D] = T2.start[D].

overlaps(T1, T2, D) :-
T1.start[D] < T2.start[D] /
T1.end[D] < T2.end[D] /
T2.start[D] < T1.end[D].

Region Connection Calculus (RCC)

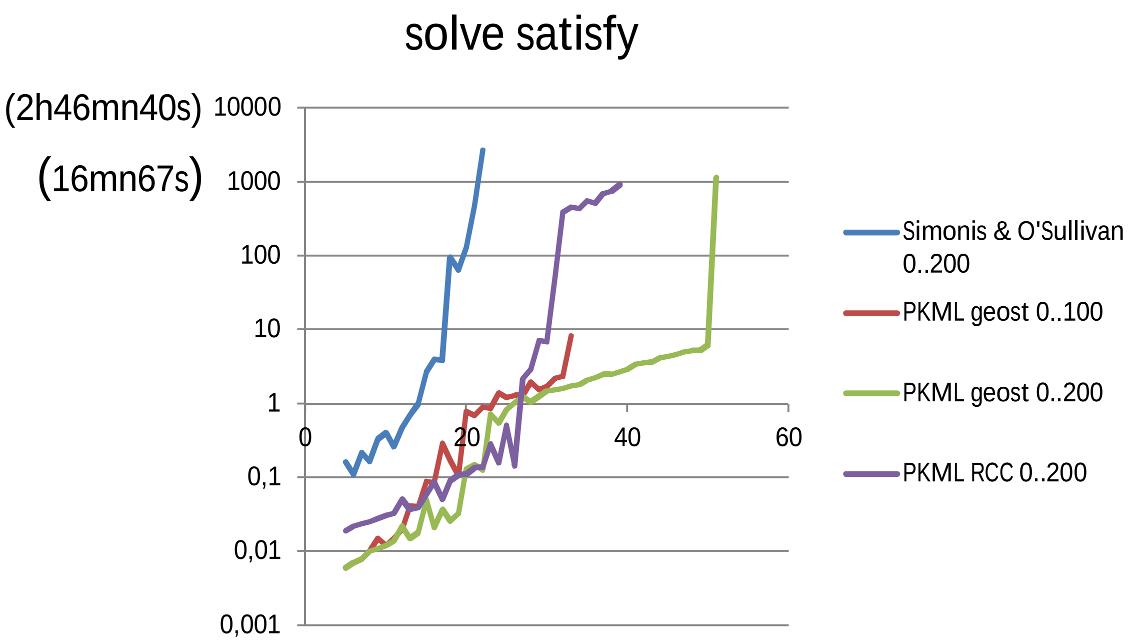
```
task(Task, Start, End) :- Task = (start: Start, end: End).

disjoint(T1, T2, N) :-
   exists(i in 1..N) ( precedes(T1, T2, i)  \/  preceded_by(T1, T2, i)).

overlap(T1, T2, N) :-
   forall(i in 1..N) ( overlaps(T1, T2, i) ).
```

Experiments: Integers

- Korf in CLPZINC using CHOCO:



- PSA in CLPZINC: 0,263s

  Gravity; Weight stacking; Weight balancing; Stack oversize.

Real numbers

- Native in MiniZinc/FlatZinc and IBEX
- Added to CHOCO parser and to CLPZINC.

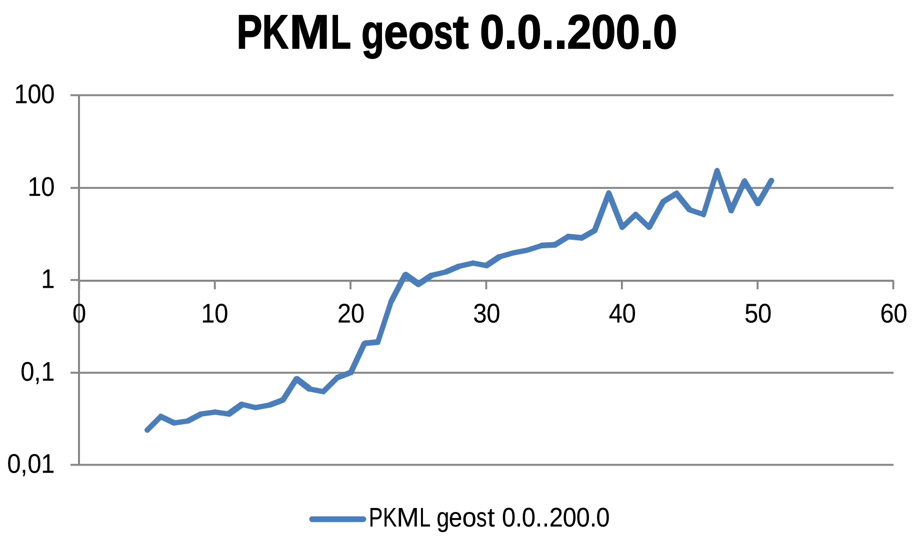
- <u>Example in CLPZINC for packing:</u> Approximating real variables on a multi-dimensional grid

```
grid(Y1, Y2, X, N) :-
        abs(Y2 – Y1) <= 1.0 / (2.0 * int2float(N)),
        Y2 = int2float(X) / int2float(N).
```

Experiments: Reals

- Korf in CLPZINC using CHOCO + IBEX:

grid = [5, 6]



Conclusion

- CLPZINC is a modeling language for expressing search strategies for CSP
- Allen, RCC and PKML are CLPZINC libraries
- Real numbers are proposed in the compilation chain
- Application to packing without significant performance loss

- <u>Future work (in another life!):</u>
- Port reals to MiniZinc v2.0 and CHOCO v3.2
- What if objects have curved shapes? e.g., circles with PKML real (CMA-ES) ...